游戏开发

华清远见-成都中心-H5教学部

第31章 CANVAS

华清远见-成都中心-H5教学部

目录

认识SVG

- 什么是SVG？
  - SVG是一种可伸缩的矢量图型，它基于XML并用于描述图形的语言；
  - 不同于用像素来描绘的矩阵图像（JPG、PNG、GIF），SVG是分辨率无关的；
  - SVG图像可以通过JS和DOM操作来创建和操控；
  - SVG有自己庞大的语法和较大的复杂度，我们这里只是了解下有这种图像格式；

canvas画布-基本概念

- canvas画布：
  - canvas本身没有任何外观，只是在文档中创建了一个画板；
  - ie9之前的版本不支持canvas；
  - 画布的尺寸要设置为属性，不要直接在css里面定义；
  - 画布的getContext()方法返回一个“绘制上下文”对象；
    - 绝大多数的画布绘制API来自这个对象；
    - 也就是说画布元素和他的上下文对象是两个完全不同的概念；
    - 调用该方法时，传递的参数是“2d”，也就是getContext('2d')，可以在画布上绘制二维图像
    - 3d绘制就相对比较复杂了，具体实现还在规范中；

canvas画布-绘制线段和填充多边形

- 绘制线段和填充多边形：
  - 绘制线段的API是上下文对象的方法；
  - beginPath：开始定义一条新的路径；
  - moveTo：开始定义一条新的子路径，该方法确定了线段的起点；
  - lineTo：将上面定义的线段起点和指定的新的点连接起来；
  - 到这里只是规划好了思路，还没有在画布上画出任何图形；
  - fill()：填充区域，此时只是填充，起点和终点并没有连接起来；
  - closePath：会把起点和终点连接起来；
  - stroke()：开始绘制图形，当前路径下的所有子路经都会绘制出来；
  - 如果要接着绘制新的路径，记得调用beginPath()方法；

canvas画布-绘制线段和填充多边形

- 实例：

```
var mycanvas = document.getElementById('mycanvas');
var c  = mycanvas.getContext('2d'); //得到一个上下文对象
c.beginPath();
c.strokeStyle = 'red';
c.moveTo(80, 120);              //起点坐标(80, 120)
c.lineTo(300, 300);             //起点坐标(80, 120) 终点坐标(300, 300)
c.lineTo(200, 100);             //起点坐标(300, 300) 终点坐标(200, 100)
c.closePath();                  //起点和终点连接起来
//再单独画一条线
c.moveTo(300, 200);
c.lineTo(300, 500);
c.fill();                       //填充上面两根线环绕的区域
c.stroke();                     //开始绘制图形
```

canvas画布-图形属性

- 图形属性：画布API给上下文对象定义了15个图形属性

属性	含义
fillStyle	填充时候的颜色、渐变或图案等样式
font	绘制文本时的字体
globalAlpha	绘制像素时候要添加的透明度
globalCompositeOperation	如何合并新的像素点和下面的像素点
lineCap	如何渲染线段的末端
lineJoin	如何渲染线段的顶点
lineWidth	线段的宽度
miterLimit	倾斜定点的最大长度
textAlign	文本水平对齐方式
textBaseline	文本垂直对齐方式
shadowBlur	阴影的清晰或模糊程度
shadowColor	下拉阴影的颜色
shadowOffsetX	阴影的水平偏移量
shadowOffsetY	阴影的垂直偏移量
strokeStyle	勾勒线段时的颜色、渐变或图案等样式

canvas画布-图形属性

- 实例：

```
var mycanvas      = document.getElementById('mycanvas');
var c             = mycanvas.getContext('2d');//得到一个上下文对象
c.beginPath();
c.strokeStyle     = 'red';      //定义填充颜色
c.lineWidth       = '3';        //定义线条粗细
c.shadowColor     = '#001';     //定义阴影颜色
c.shadowBlur      = '10';       //定义模糊程度
```

canvas画布-画布的尺寸和坐标

- 画布的尺寸和坐标：
  - 画布以左上角(0, 0)为坐标原点；
  - 往右为X轴的坐标不断增大；
  - 往下为Y轴的坐标不断增大；
  - translate()：坐标上下左右移动；
  - rotate()：将坐标轴根据指定角度顺时针旋转；
    - 需要注意的是画布API是根据弧度表示角度的； 弧度 = ((2π*半径/360)*角度)/半径 = 角度*Math.PI/180;
  - scale()：对X轴或Y轴的距离进行延长或缩短； 传递负值时，实现以坐标原点为参照点，将坐标轴进行翻转；

canvas画布-曲线的绘制和填充

- arc()：在当前子路经添加一条弧线；
  - 六个参数：
    - 圆心坐标x,y：和moveTo指定的坐标不同；
    - 半径r：圆心到弧形的距离；
    - 弧形起始和结束弧度：水平向右是0，垂直向下是90，水平向左是180；
    - 弧形方向：顺时针false，逆时针true，默认false；

- 实例：

```
var mycanvas      = document.getElementById('mycanvas');
var c             = mycanvas.getContext(‘2d’);//得到一个上下文对象
c.lineWidth       = 1;
c.beginPath();
c.moveTo(100, 100);
//圆心坐标  半径 开始弧度  结束弧度
c.arc(100, 100, 80, Math.PI*3/2, Math.PI*323/180, 0);
c.fill();
c.stroke();
```

canvas画布-矩形的绘制和填充

- 矩形的绘制和填充：
  - rect()：在当前子路径添加一条弧线；
  - 四个参数：
    - 起点坐标x,y：左上角坐标； 宽度width：矩形的宽度； 高度height：矩形的高度；
  - strokeRect()方法可以直接绘制一个矩形；
  - 实例：

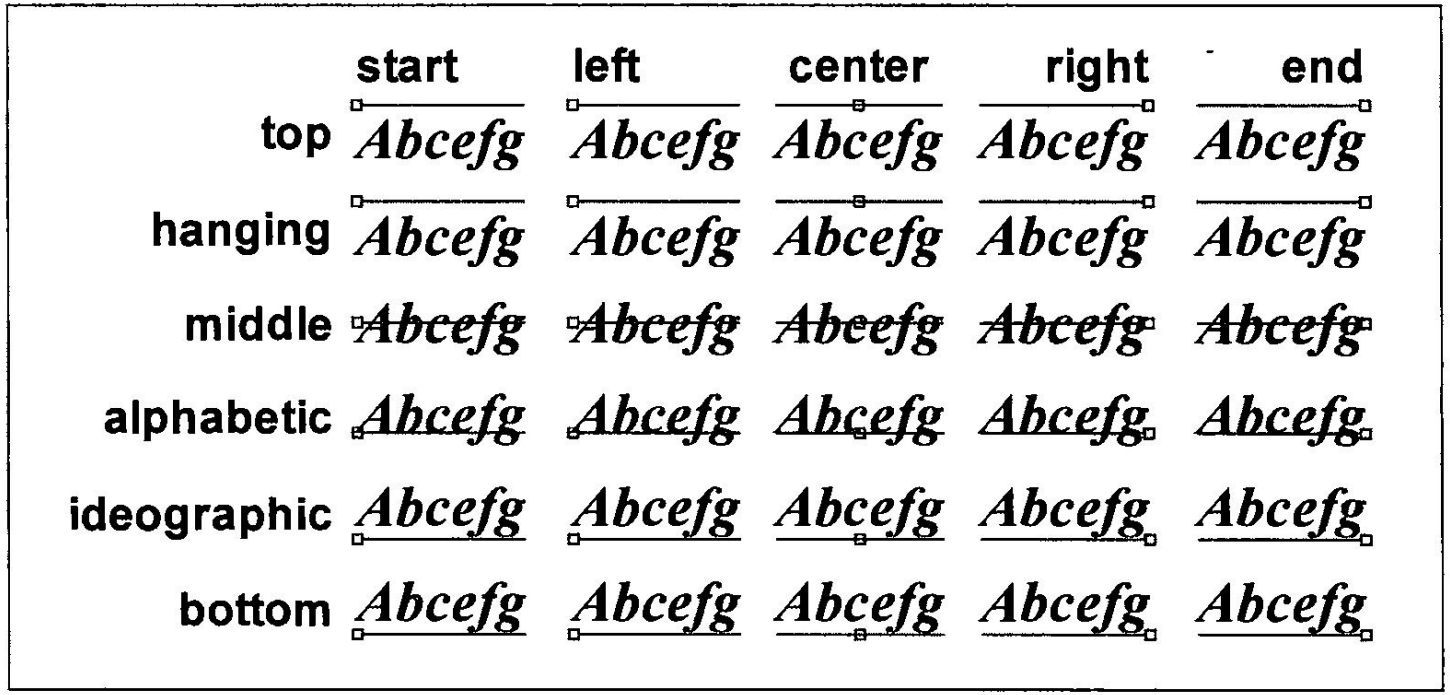
```
var c = mycanvas.getContext('2d');//得到一个上下文对象
c.rect(100, 10, 180, 100);
c.fill();
c.stroke();
//画个矩形
c.lineWidth   = 1;
c.strokeRect(100, 120, 200, 100);
```

  - clearRect()方法清除矩形内的内容；

```
c.clearRect(130, 90, 200, 60);
```

canvas画布-绘制文本信息

- fillText()：绘制文本信息：
  - 三个参数： 绘制的内容； 起点x坐标； 起点y坐标；
  - 文本颜色使用fillStyle属性指定；
  - 文本字体使用font属性指定，和CSS一致；
  - textAlign属性指定水平方向对齐方式，可选值：start、left等，textBaseline则指定垂直方向，可选值：top、hanging等，参考下图：



canvas画布-图片

- 图片：
  - drawImage()：将原图片像素的内容复制到画布上；
  - 第一个参数是源图片，可以是img元素或Image构造函数创建的屏幕外图片对象；
  - 三个参数时： 指定图片绘制的x、y坐标；
  - 五个参数时： 指定图片绘制的x、y坐标，以及图片的宽度、高度；
  - 九个参数时： 裁剪的对象 裁剪的位置（x,y） 裁剪的宽度和高度(w,h) 裁剪后图片绘制的位置(x,y) 图片显示出来的宽度和高度(w,h)

canvas画布-图片

- 实例：

```
var mycanvas        = document.getElementById('mycanvas');
var c               = mycanvas.getContext('2d');//得到一个上下文对象
var img             = new Image();
img.src             = './img/header.jpg';
//要等待图片加载完成后才开始绘制图片
img.onload = function () {
        //绘制原始图片：3个参数
        c.drawImage(img, 0, 0);
        //规定图片绘制大小：5个参数
        c.drawImage(img, 100, 100, 200, 200);
        //裁剪的时候9个参数要完整
        c.drawImage(img, 20, 30, 50, 50, 300, 300, 50, 50);
}
```

第32章 大数据可视化

华清远见-成都中心-H5教学部

大数据可视化

目录

平台选择

数据可视化的插件还是比较多，比如Highcharts、echarts、D3等，这里我们选择百度的echarts。

官网地址：https://echarts.apache.org/zh/index.html

模块引入

需要下载源代码或者npm安装到本地，更多的可参考文档：https://echarts.apache.org/zh/tutorial.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <!-- 引入 ECharts 文件 -->
    <script src="echarts.min.js"></script>
</head>
</html>
```

图表绘制

1，准备一个具备高宽的盒子： ；

2，初始化数据：

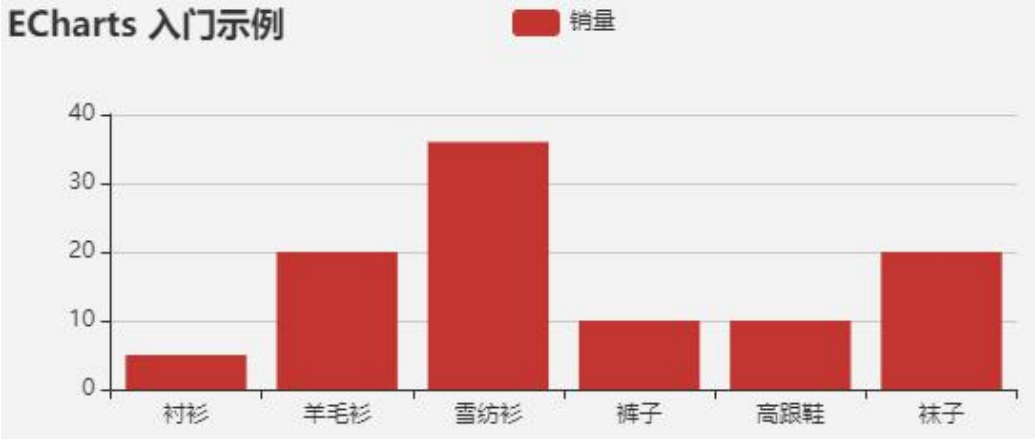
```
<script type="text/javascript">
    // 基于准备好的dom，初始化echarts实例
    var myChart = echarts.init(document.getElementById('main'));

    // 指定图表的配置项和数据
    var option = {
        title: {
            text: 'ECharts 入门示例'
        },
        tooltip: {},
        legend: {
            data:['销量']
        },
        xAxis: {
            data: ["衬衫","羊毛衫","雪纺衫","裤子","高跟鞋","袜子"]
        },
        yAxis: {},
        series: [{
            name: '销量',
            type: 'bar',
            data: [5, 20, 36, 10, 10, 20]
        }]
    };

    // 使用刚指定的配置项和数据显示图表。
    myChart.setOption(option);
</script>
```

图表绘制

就这么简单，你的第一个图标就诞生了。



异步数据的加载和更新

很多时候数据需要异步加载后显示。ECharts 中实现异步数据的更新非常简单，在图表初始化后，通过AJAX异步获取数据后通过 setOption 填入数据和配置项就行了。

具体参考官方文档：

https://echarts.apache.org/zh/tutorial.html#%E5%BC%82%E6%AD%A5%E6%95%B0%E6%8D%AE%E5%8A%A0%E8%BD%BD%E5%92%8C%E6%9B%B4%E6%96%B0

第33章 本地数据存储

华清远见-成都中心-H5教学部

目录

客户端存储-概念及概述

- 客户端存储：
  - web应用允许使用浏览器提供的API将数据存储在客户端电脑上；
  - 客户端存储遵守“同源策略”，不同的站点页面之间不能相互读取彼此的数据；
  - 在同一个站点的不同页面之间，存储的数据是共享的；
  - 数据的存储有效期可以是临时的，比如关闭浏览器数据就销毁； 也可以是永久的，可以在客户端电脑上存储任意时间；
  - 在使用数据存储是需要考虑安全问题，比如银行卡账号密码；

localStorage和sessionStorage

- localStorage和sessionStorage是window的两个属性
  - 他们代表同一个Storage对象；
  - 可以存储字符串类型的数据：

    localStorage.sname = '华清远见成都中心';

    console.log(localStorage.sname);//其它页面可以获取该缓存数据

  - 存储数组数据：对结构化的数据存储时编码，使用时解码

    localStorage.myarr = JSON.stringify([1,2,3,4,5]);

    console.log(JSON.parse(localStorage.myarr));

  - 存储对象类型的数据时同上；
  - 存储日期类型的数据：

    localStorage.mydate =(new Date()).toUTCString();

    console.log(new Date(Date.parse(localStorage.mydate)));
  - 以上存储的数据在其他页面都是可以直接获取的；

localStorage和sessionStorage

- localStorage和sessionStorage存储数据的有效期和作用域不同：
  - localStorage：存储的数据是永久的，当然用户可以清除这些缓存数据的； 比如清除历史记录就可以清除这些数据；
  - localStorage的作用域限制在文档源的； 文档源由协议、主机名、端口三者来确定；下面的URL就拥有不同的文档源：
    - http://www.hqyj.com //协议http，域名www.hqyj.com
    - https://www.hqyj.com //协议https，和上面不同
    - http://www.farsight.com.cn/ //和上面域名不同
    - http://www.farsight.com.cn:81/ //和上面端口不同，默认端口是80
  - 这种情况下，localStorage存储的数据是不能相互访问的； 即便他们来自同一台服务器；
  - localStorage同源的文档之间可以相互访问和修改相同名称的数据；
  - localStorage受浏览器厂商的限制，chrome下存储的数据，360浏览器下不可访问； 会得到‘Invalid Date’；

localStorage和sessionStorage

- localStorage和sessionStorage存储数据的有效期和作用域不同：
  - sessionStorage存储的数据在窗口或标签关闭时，数据就会丢失； 在一个标签前进后退时数据不会丢失，这样我们就可以获取上次访问时产生的相关信息； 随着现代浏览器越来越强大，具体的声明周期还和浏览器功能有关；
  - sessionStorage在localStorage的同源策略基础上，有更严格的限制：
    - 他还被限制在窗口中，意思是同一个窗口或标签页的不同页面之间可以共享sessionStorage；
    - 但是不同的窗口或标签页之间不能共享sessionStorage，即便他们是同一个页面地址；
    - 窗口是指顶级窗口，如果是多个iframe，他们之间共享sessionStorage；

localStorage和sessionStorage

- localStorage和sessionStorage的API：
  - localStorage和sessionStorage可以当做普通的API，但同时也提供了正式的API，操作起来更加方便；
  - setItem()：将对应的名字和值传递进去，可以实现数据存储；
  - getItem()：将名字传递进去，可以获取对应的值；
  - removeItem()：将名字传递进去，可以删除对应的值；
  - clear()：删除所有的缓存值，不需要参数；
  - length：属性，获取键值对总数；
  - key()：传入位置数，获取存储的值的名字；

localStorage和sessionStorage

- localStorage和sessionStorage的存储事件：
  - 存储在localStorage和sessionStorage上的数据发生改变时， 浏览器会在其他对该数据可见的窗口对象上触发存储事件； 对数据进行改变的窗口不会触发该存储事件；
  - 如果两个标签页打开了同源页面，那么其中一个在localStorage存储数据时，会触发另一个标签页的存储事件；sessionStorage的作用域确定了他没有这个特性；
  - 只有当存储数据真正发生改变时，才会触发存储事件； 删除不存在的或给存储项设置一样的值不会触发存储事件；
  - localStorage和存储事件采用广播机制，浏览器会对正在访问同样站点的所有窗口发送消息；

localStorage和sessionStorage

- localStorage和sessionStorage的存储事件：
  - 与存储事件相关的事件对象的属性：
    - key：被设置或移除的键名；
    - newValue：该项的新值，如果是删除返回null；
    - oldValue：改变或删除选项前的值；添加新值时为null；
    - storageArea：类似window对象上的localStorage或sessionStorage；
    - url：触发事件的脚本URL；
  - 实例：需要虚拟路径的方式访问

```
document.getElementById('myinput').onkeyup = function (argument) {
        localStorage.setItem('myname', this.value);
}
window.addEventListener('storage', sto);
function sto () {
        document.getElementById('myinput').value = localStorage.getItem('myname');
}
```

cookie

- Cookie可以实现本地数据存储：
  - 创建cookie：
  - document.cookie = 'age=20'；
  - 指定有效期：ocument.cookie = 'name=刘桐;max-age=' + 720*3600；

- 获取cookie信息：

  document.cookie

localStorage、sessionStorage、cookie

- 三者的区别：
  - 存储大小
    - cookie数据大小不能超过4k ；
    - sessionStorage和localStorage 虽然也有存储大小的限制，但比cookie大得多，可以达到5M或更大；
  - 有效时间
    - localStorage 存储持久数据，浏览器关闭后数据不丢失除非主动删除数据；
    - sessionStorage 数据在当前浏览器窗口关闭后自动删除；
    - cookie 设置的cookie过期时间之前一直有效，即使窗口或浏览器关闭；
  - 数据与服务器之间的交互方式
    - cookie的数据会自动的传递到服务器，服务器端也可以写cookie到客户端
    - sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存；

localStorage、sessionStorage、cookie

- 作用域
  - localStorage的作用域限制在文档源的；
  - localStorage同源的文档之间可以相互访问和修改相同名称的数据；
  - localStorage受浏览器厂商的限制，chrome下存储的数据，360浏览器下不可访问； 会得到‘Invalid Date’；
  - sessionStorage在localStorage的同源策略基础之上，还有更严格的限制:
    - 他还被限制在窗口中，意思是同一个窗口或标签页的不同页面之间可以共享sessionStorage；
    - 但是不同的窗口或标签页之间不能共享sessionStorage，即便他们是同一个页面地址；
    - 这里的窗口是顶级窗口，如果里面有多个iframe，他们之间共享sessionStorage；

海量视频 贴身学习

超多干货 实时更新

THANKS

谢谢